The Benefits of ODR in Complex Software Contract Disputes

Using the *Forensic Systems Analysis* methodology to arrive at and present an expert opinion for Online Dispute Resolution purposes

by

Dr Stephen Castell*
Chartered IT Professional

Chairman, *CASTELL Consulting*
Member of the Expert Witness Institute
Committee Member, British Computer Society *Law Specialist Group*

Paper presented at the ***Third Annual Forum on Online Dispute Resolution***, hosted by the International Conflict Resolution Centre at the University of Melbourne, Australia, 5-6 July 2004, in collaboration with the United Nations Economic and Social Commission for Asia and the Pacific (UNESCAP); Day 2, Tuesday 6 July 2004, Workshop Streams 14.00-15.30, '**Applications of ODR – *Commercial Disputes***'.

Summary

Software implementation contracts are frequently *terminated* with the software *rejected* amidst allegations from both supplier and customer, e.g. *software/database errors/deficiencies*, *faulty design*, *shifting user/business requirements*. An important technical issue on which the IT Expert appointed in such disputes is asked to give an expert opinion is: *what was the quality of the delivered software and was it fit for purpose?* ODR can bring benefits in presenting the objective findings of the IT Expert through online demonstration of the results of *Forensic Systems Analysis*, providing dramatic insight into the state of the software, saving time and costs, and facilitating settlements.

Introduction – Features of IT Disputes

Those who have been involved in litigious disputes over failed computer software projects would readily agree that, whatever their size in terms of the financial amounts at stake (and whatever the facts and circumstances of the contract between the parties, and the conduct of the subsequent software development) software construction and implementation cases present interwoven technical and legal issues which can be both arcane and complex – and therefore prove costly and time-consuming to unravel.

CASTELL Consulting, an independent professional IT consultancy, founded in 1978, has been involved in a wide variety of such complex computer software litigation **[1]**. We have in particular been instructed as expert witnesses (in e.g. the UK, Europe, the Arabian Gulf, Australasia, the USA) in many legal actions concerning major software development contracts which have been *terminated*, with the software *rejected* amidst allegations of *incomplete* or *inadequate delivery*, *software errors*, *shifting user specifications*, *poor project management*, *delays* and *cost over-runs*. This work has been on behalf of Claimants and Defendants, software customers and suppliers, in the High Court (or equivalent), Arbitration, Mediation and other forms of ADR. In addition, I have personally acted as an ICC Arbitrator, and CEDR-trained Mediator; and as *Technical Assessor* to an International Arbitrator (in a Hong Kong case for tens of millions of US$), in a role similar to that which, under the recently introduced English High Court *Civil Procedure Rules*, has become increasingly familiar to many UK IT expert witnesses – being appointed *Single Joint Expert* ('*SJE*').

Forensic Systems Analysis

Over the years of examining mixed and varied software development disputes as appointed experts, *CASTELL* has developed a range of techniques for assessing and reading the 'technical entrails' of failed, stalled, delayed or generally troublesome software development projects. It should be noted that such projects can these days often be a contractually uncertain mixture of 'customised' software packages *and* 'bespoke' construction. Many articles and papers have been written as a result of such experiences **[2]**.

This *CASTELL* inquisitorial method, *Forensic Systems Analysis* **[3]**, focusing as it does on *testing of the software in dispute*, is I believe capable of being developed into a protocol for presenting the objective findings of the IT Expert in the context of Online Dispute Resolution. I believe that, through online demonstration of the results of *Forensic Systems Analysis*, the IT Expert can provide illuminating and dramatic insights into the state of the software in dispute, and contribute mightily to saving time and costs in, and facilitating rapid settlement of, such technically complex disputes.

Furthermore, such techniques of Online Dispute Resolution and *Forensic Systems Analysis* could, I suggest, be used not merely in a dispute context, but also to assess the fragile status and troublesome characteristics of specific 'problem software projects/contracts' *before* they stall, fail, or sink into litigation; and, more generally, as a positive and rigorous 'litigation sensitive' *Software Quality Assurance and Project Management Audit Method* for large software construction and implementation projects, throughout their conduct. I do not, however, explore these further aspects and benefits in this short paper.

I here outline just some of the *Forensic Systems Analysis* components relevant to expert investigation in typical civil litigation over failed software development contracts, the application and findings of which could, I believe, with great benefit be presented online for Online Dispute Resolution purposes. [Further details will be available on a future website *www.ForensicSystemsAnalysis.com*].

Software 'quality'

The most common, and arguably most important, issue on which the computer expert is inevitably asked to give a view in software development or implementation cases is: *what was the quality of the delivered software and was it fit for purpose ?* This raises the question of: *just what is meant by 'software quality'* ? The ready answer from the experienced IT expert is that 'quality' can only mean 'fitness for purpose', in the sense of '*does the delivered software meet its stated requirements ?*. Thus:

(1) 'Quality' of software is a concept which is essentially dependent on the specification of what the software is expected to do and how the software is expected to perform in its defined environments. In other words, the yardstick for measuring and judging whether software is of appropriate quality and fit for its intended purpose is the ***Statement of Requirements*** *defining what is required or expected of it; and*

(2) ***Testing software against its Statement of Requirements*** *is the* ***only*** *practical and universally accepted method of judging the quality of the software and whether or not the software is fit for its intended purpose.*

This critical focus on *testing the software in dispute against its Statement of Requirements* has a different emphasis for different specific cases.

*For example, in a case concerning an in-store EFTPOS system for a major national retailer, the crucial issue was whether or not the software supplier was likely to have fixed many outstanding errors and have had the system ready to roll-out in time for the pre-Christmas sales rush. What was the objective technical evidence of the software house's '****bug find and fix****' performance ? Were the bugs escalating, or was the software converging onto a stable, performant system ? Were, rather, the constant changes in customer specification – as alleged by the supplier – perhaps to blame for the delays and inability of the software to pass a critical acceptance test ?*

*A case concerning a large University Consortium similarly focused on the apparent inability of the software developer to present a main module of the software system in a state capable of passing formal Repeat Acceptance Tests, with a number of faults appearing at each attempt at 'final' testing (even though three earlier main modules had been successfully developed and accepted). How serious were these faults, and were earlier faults that had been thought to have been fixed constantly re-appearing ? Was the customer justified in terminating the contract on the grounds of a '**reasonable opinion**' that the software supplier would not resolve all the alleged faults in a 'timely and satisfactory manner' ? Was the supplier's counter-claim for a large financial amount for '**software extras**' valid, and could that explain the inability of the software to converge onto an 'acceptable' system ?*

*In another case – that of a real-time computer-aided mobilising system for a large ambulance brigade – the focus was on the **response times** of the software in a clearly life-or-death application. How well were the desired response, availability, reliability and recovery targets for the software contractually defined, and what was the evidence of the system's actual **performance** under varying load conditions ?*

Testing Incident Reports

The computer expert witness – often coming onto the scene of the failed project many months, sometimes years, after it has all collapsed – is usually presented with large volumes of project documentation, an important element of which is the set of *software testing records*. Typically, these are in the form of *Testing Incident Reports* ('*TIRs*'), and they can run into many hundreds, if not thousands or tens of thousands, for large-scale bespoke software development contracts.

To simplify, the dispute may then come down to this. The customer alleges that the *TIRs* represented *errors* in the software which were critical, serious, incapable of being remedied, too numerous in number, or in some other way, or ways, either were, or summed up to, a *material breach* of the contract by the software supplier, entitling the customer to reject the software and terminate the contract. The software vendor/developer, on the other hand, retorts that the *TIRs* did not constitute 'showstopper' faults, they were readily technically rectifiable, and anyway principally arose from the many and continuous changes in specification made by the customer – the customer was *not* entitled to terminate and had himself *repudiated the contract* in so doing.

The *Forensic Systems Analysis* Methodology: *EFLAT*, *EAT* and *FORBAT*

The expert hired by either of the parties in the dispute (or as an *SJE*) may address these issues using a number of *Forensic Systems Analysis* components, the most important of which, for the purposes of realising the presentational benefits of ODR, are likely to be ***EFLAT***, ***EAT*** and ***FORBAT***, outlined as follows.

EFLAT – Expert's Fault Log Analysis Task – Material Defect

During software development defects are routinely encountered, and routinely fixed, and there is generally nothing alarming about their occurrence. For the purposes of *rejection* of software and *termination* of a software development contract, any alleged defect must therefore be assessed using a strict test as to whether or not it is truly a ***material defect***, that is, as to whether or not '***the contract cannot be considered to have been performed while this defect persists***'.

EFLAT, developed over the years through careful debate with many firms of instructing solicitors, and learned Counsel, uses what I believe is a sound protocol for testing whether or not any given software fault, in terms of its relevance to a breach and termination of a contract, is truly a ***material defect***.

This protocol is essentially that, to be a *material defect*, an alleged software fault must be

(1) of large consequential (business) effect; ***and***
(2) impossible, or take (or have taken) a long time, to fix; ***and***
(3) incapable of any practical workaround.

The customer is quite properly entitled to define what is a 'large' consequential business effect; and the supplier, equally, may put forward an appropriate sizing for a 'long' time to fix – each from the standpoint of his own business/technical knowledge and experience, and in the context of the particular contract/project. Both views ought to be evidentially supportable. Both views – and, also, whether or not there *is* indeed a practical workaround – would be the subject of expert scrutiny and opinion.

EFLAT constitutes a careful re-running of the appropriate Acceptances Tests, under expert observation, with each *TIR* (or '*Fault Log*') raised during the test rigorously and dispassionately assessed according to the *material defect* rule. The outcome is a *Scott Schedule (of Software Defects)* with each fault particularised, stating why each was considered a breach of contract (by reference to specific contractually defined requirements), what the consequential effect was estimated to be, what the technical time to fix was (or was estimated to be), whether or not there was any practical workaround available; giving the expert's independent view on all these individual elements, with, finally, an opinion as to whether or not the specific fault in total was a *material defect*. A pro-forma for the *Scott Schedule (of Software Defects)* is given at **Annex A** hereof, and such a pro-forma is, I believe, readily adaptable to being presented, for example via (updatable) web pages, in the context of Online Dispute Resolution.

EFLAT is undertaken with a 'prototyping' orientation, assessing first only a limited proportion of the *TIRs*, so that experience may be gained as to how difficult the full task is likely to be, how long all the *TIRs* will take to assess, and whether there may be technical obstacles in, say, reproducing the exact conditions of the Acceptance Test corresponding to those which obtained when the alleged faults were originally found. Not the least of these obstacles can be the basic *evidential uncertainty* over whether or not the *version* of the applications software and/or the database *configuration* and/or the hardware and systems software *environment* available to the expert (months or years later) precisely correspond to the system being litigated over.

Obstacles apart, early prototyping of ***EFLAT*** enables estimates of how much time (and therefore cost) is likely to be needed to complete the *Scott Schedule (of Defects)* in its entirety, enabling clients and instructing solicitors to take a considered view as to the full extent of expert investigation to be commissioned. Once again, such estimates, properly presented and shared through e.g. controlled web 'publication', should be of great benefit in the conduct and costing of an Online Dispute Resolution.

EAT – 'Extras' Analysis Task

It can be that, in software development projects of any significant size, there are many 'contract variations' caused by the inevitable shift in the customer's or users' perceptions of what they require as, for example, they see the software actually being built and tested. This 'specification drift' or 'constant changes to requirements' is a well-known phenomenon in almost all engineering construction disciplines and presents a particular challenge to well-ordered project management, to ensure that such variations are at all times properly documented and controlled, and that both parties understand and agree the impact on project scope, timetable and costs which implementing all requested software changes could have.

Typically, for the software systems project which collapses and ends in dispute or litigation, the computer expert witness is asked to give opinion on whether or not there were indeed changes from the originally contracted software; and, if so, what was the quality of the additional software built; and to what financial remuneration (e.g. on a *quantum meruit* basis) the supplier may be entitled for providing such software 'extras'.

EAT comprises a methodical analysis of (1) the *contractual documentation* (in particular the *Statement of Requirements*, including any amendments or re-issues thereof during the project); (2) the *work records* of the software engineers who did the construction of the 'extras'; (3) the items of software *design, source code, functionality, execution* and *performance* which it is alleged have been produced as a result of all this extra work; and (4) the financial amount claimed, and if it is consistent with (1)-(3) and passes a 'sanity cross-check' such as that provided by assessing the "£ or $ per delivered line of source code' standard software metric.

A pro-forma for the resulting *Scott Schedule (of Software Extras)* is given at **Annex B** hereof. Once again, a 'prototyping' methodology is used to give client and instructing solicitors an early reading as to the likely time and costs needed to reach a complete opinion on all items of software 'extras' claimed. And, once again, such a pro-forma and its associated insights are, I believe, readily adaptable to being presented with great benefit in the context of Online Dispute Resolution.

FORBAT – FORensic Bug Analysis Task

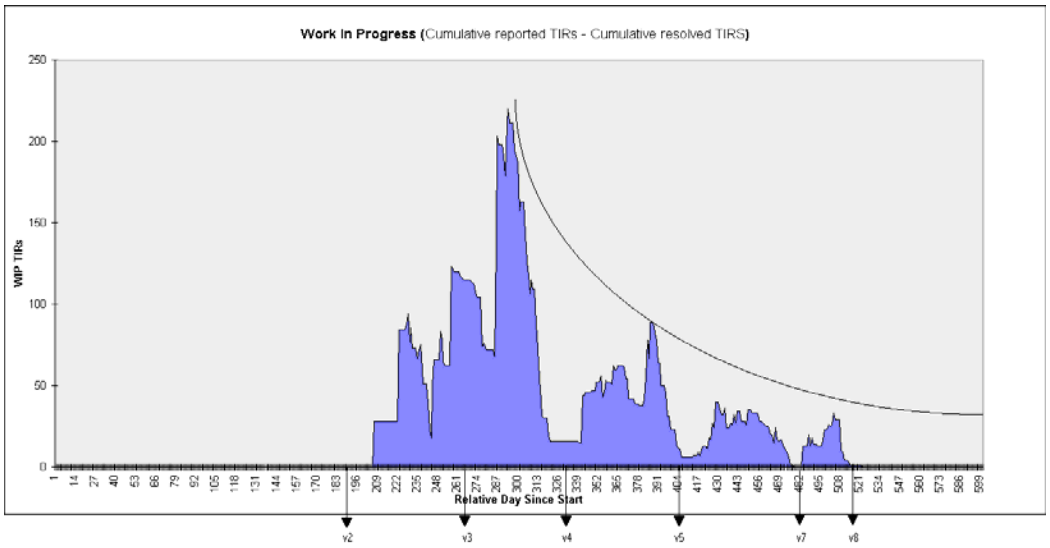
Always recognising that during software development defects are routinely encountered, and routinely fixed, and there is generally nothing alarming about their occurrence, the overall numbers of such 'bugs' (as logged by the *TIRs*), and the pattern of their build-up and resolution, are nevertheless important indicators of the progress of software construction and testing.

Such indicators are unfortunately often misread by both the software customer and the software developer: in particular, the dramatic increase in *TIRs* and apparent 'never-ending increase in bugs' during systems testing can be badly misinterpreted. The point is that *systems testing* (usually the responsibility of the software developer) is *meant* to find bugs and fix them – it is not being done properly if there is not a large build-up in recorded *TIRs*. This contrasts with *acceptance testing* (usually the responsibility of the customer) where 'zero', or only a small number of non-serious bugs, is a not unreasonable expectation, particularly as acceptance testing should be undertaken with the appropriate attitude - for *acceptance*, not *rejection* of the software proffered for testing.

FORBAT uses a number of standard *quantitative* analysis techniques to give an objective graphical presentation of the true 'bug find and fix' performance of the software house, readily understandable, with a little explanation, to non-technical clients, lawyers or judges. The insights which spring out of these presentations are usually vivid (and incidentally can come as something of a surprise to the parties themselves). These are best explained by two examples, both taken from a real software project:

Illustrations of typical FORBAT 'bug find and fix performance' graphs; and conclusions reached

Graph 1: "Work in Progress"



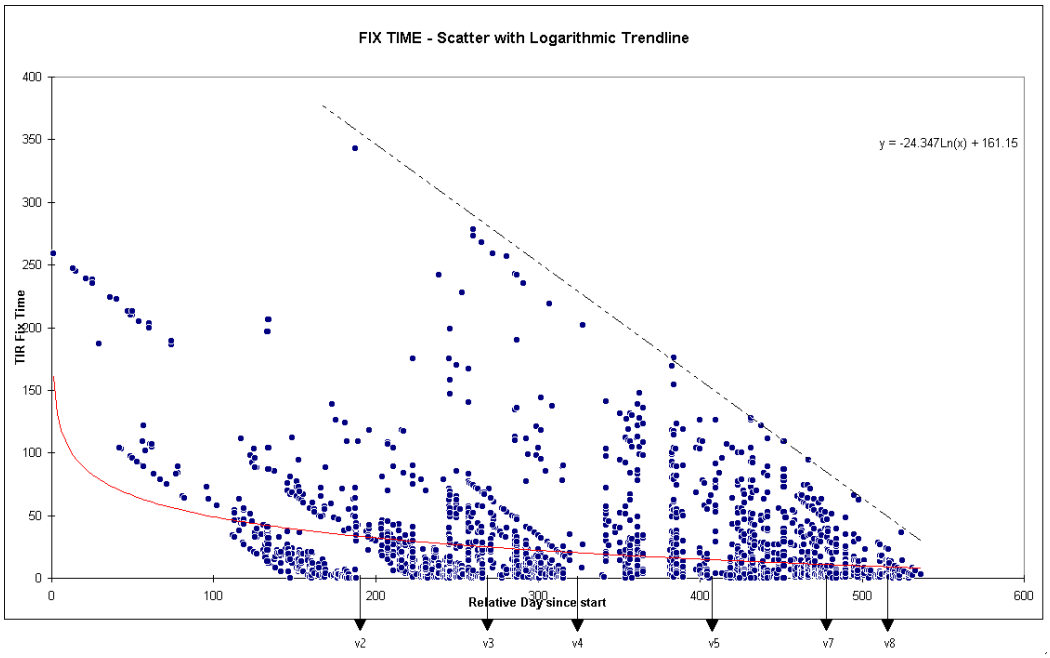
"Graph 1 represents the cumulative number of *TIRs* outstanding at any point in time, and is formed by subtracting the cumulative total number of resolved *TIRs* from the cumulative total number of reported *TIRs* on a day-by-day basis. A trendline has been added to the graph to indicate the overall tendency of the underlying data, but otherwise the graph is an objective portrayal of basic project statistics. From this graph it can be seen that the number of outstanding *TIRs* peaked at more than 200 before the release of Version 4, but that these outstanding *TIRs* had been addressed and resolved, with the exception of fewer than 20, by the time Version 4 was released. This is in keeping with what would be reasonably expected if the software house were efficiently handling the 'errors' reported, eliminating

as many bugs as possible before releasing a new version of software. The steep drop-off in the number of outstanding *TIRs* from day 300 to day 320 is an indication of a significant amount of work performed to address and resolve outstanding *TIRs* prior to release of a new version of the software. The inferences to be drawn from this graph and its underlying data are: for each major release of the software (i.e. Versions 4, 5, 7 and 8):

1) the total cumulative number of outstanding *TIRs* decreased steadily from more than 200, to less than 100, to less than 50 to around 45;
2) the total number of outstanding *TIRs* reduced sharply before each software release (notice the steep drops in the curve before each major release, indicating a 'clean-up' process prior to each release);
3) the total number of outstanding *TIRs* for each major release decreased from less than 20, to less than 10, to less than 5, to zero.

These results are consistent with what is reasonably to be expected in a well-run software development project environment. It is usual in such an environment to see a build-up of reported 'errors' or 'bugs' during systems testing as a major release approaches. It is also usual to see a steep drop-off in the number of these bugs outstanding just before each major release. In summary, Graph 1 illustrates that the process to 'find and fix bugs' was one in which the software house became increasingly proficient; and indicates that the software itself was becoming more and more stable."

Graph 2: "FIX TIME: Scatter with Logarithmic Trendline"



"Graph 2 depicts all *TIRs* in scatter format. Each individual *TIR* is plotted on this Graph at the point in time that it was reported versus the elapsed time in days that it took to resolve it. A logarithmic trendline has been calculated based upon the distribution of the base data used, with the resulting equation indicated in the upper right hand portion of the Graph and the trendline itself superimposed on the data. This trendline indicates an overall tendency of the data points plotted. Additionally, a 'worst case trendline' has been drawn on this Graph (the dotted straight line running from the upper left hand side of the diagram to the lower right hand side). This 'worst case trendline' uses the outlying data points to provide an indication of where the trend is going from the 'worst case' perspective.

Looking at the 'worst case trendline', its steep slope may be noted. It appears that it would intercept the X-axis near project day 550, or 1½ months after the project was terminated. This extrapolated interception, which would have constituted a modest further elapsed time in the context of a software project which had already been running for over 18 months, corresponds to the resolution of all the longest outstanding *TIRs* within the data population."

I believe that it can be easily appreciated that the vivid and self-evident insights that spring out of such objective graphical portrayal of factual and quantitative software project analyses could with great advantage be readily promulgated and understood within the conduct of an Online Dispute Resolution.

POLIT – Performance, Operability and Locking Investigation Task

Even when the functionality provided by the software is acceptable, litigious allegations can be made over its run-time performance, perhaps expressed as 'response times did not meet those stipulated in the contract'. To form an expert view, a *system sizing and performance model* may be constructed.

POLIT uses such a model to establish *inter alia* a *resource usage profile* for key transactions used in testing scripts in order to:

(a) give a measure of the likely performance of a full system workload
(b) identify ways in which performance could be optimised or improved.

Critical resources investigated are:

(i) disk accesses and
(ii) cpu usage.

Key variables examined include:

* Resources used by log-in processes
* Paging-in of the application software the first time it is used
* Impact of different cache sizes
* Impact of different batch sizes
* Overhead of menu navigation when repeating the same process
* Effect of spreading the database
* Effect of shadowing (using two identical copies of the database).

It may be glimpsed that ***POLIT*** can become deeply technical and abstruse, and it is a challenge to the expert to explain the methodology, the model and the conclusions it delivers in a way understandable, for example, to lawyers or to the court. However, when that challenge has been met, the findings arising can also, once again, be readily shared with all those involved in an Online Dispute Resolution, by way for example of interactive web page presentation. Such presentation may perhaps even allow those involved to carry out their own 'what if' exercises using the expert system sizing and performance model provided.

Some further *Forensic Systems Analysis* tasks, the findings of each of which are I believe readily adaptable to being used in Online Dispute Resolution, are – briefly – as follows:

FUDDER – FUndamental Database and DEsign Review

Often there are allegations of 'fundamental software design flaws': this task assesses a range of accepted software engineering design parameters, and their documentation, to give an opinion on the completeness, correctness and robustness of the software, database and communications architectures in terms of their suitability for building, testing and implementing a system meeting all required contractual obligations.

PROMADET – PROject Management And Delay Examination Task

'Slippage' in project schedules is extremely common in large software development projects. The expert is often asked to answer the question: "Who was to blame for the delays ?". This task examines GANTT charts and the like, drawn and re-drawn throughout the project, together with associated project management documentation (e.g. Minutes of Project Board/Committee Meetings) and work records, as well as the 'fossil record' of the evolution of the construction of the software source code itself.

It should be noted that it can be difficult to 'win a case only on an allegation of delay': the very meaning of 'delay', and the evidence and reasons for its occurrence, are usually not easy to determine, or present clearly, for any software project which has gone on for several years. Very often the best that can be said is only that 'everyone was equally responsible' – not a particularly helpful opinion for an expert to give !

EVOCRAT – EVOlution of Changes to Requirements Analysis Task

As I have said, constant contract variation caused by 'specification drift' is a common experience in most sizeable software development projects. The expert is asked to give his view as to what all such changes amount to in terms of 'What was the final *Statement of Requirements* – what was the detailed contractual specification at the end of this process of change ?'. ***EVOCRAT*** assesses all documents purporting to state new or changed requirements, and consolidates them into a unified specification.

Conclusions

Given that the key to arriving at a useful and helpful expert opinion in complex software contract cases is *testing of the software in dispute*, I believe that the techniques and findings of the CASTELL *Forensic Systems Analysis* methodology can readily be applied to resolving disputes in the context of Online Dispute Resolution.

If the parties involved in the ODR can agree to apply these rigorous techniques they can clearly create a growing corpus of shared objective results, using a process which could perhaps be set out along the following lines:

1. Where technically possible, make the software in dispute accessible via the web.

2. Publish/disclose appropriate User Guides.

4. Experts on both sides design agreed test plans, scripts, expected results – publish/disclose them.

3. Once the *EFLAT* and *EAT* tasks based on such agreed plans etc have been undertaken, publish/disclose the relevant findings in Scott Schedules in standard/agreed form (see e.g. ***Annex A*** and ***Annex B*** hereof).

4. Further tests can then be done, and results and expert inferences/opinions added to the Scott Schedules – e.g. via hotlinks.

I believe that the benefits of such an approach in complex software contract disputes, presenting the objective findings of the IT Expert in the context of Online Dispute Resolution, could be substantial. Through online presentation, demonstration and accretion of the results of *Forensic Systems Analysis*, the IT Expert can provide illuminating and dramatic insights into the state of the software in dispute, and contribute mightily to saving time and costs in, and facilitating rapid settlement of, such technically complex disputes.

***Dr Stephen Castell, Chartered IT Professional, Chairman of *CASTELL Consulting* (founded 1978), is an internationally acknowledged independent computer expert. As an IT Expert Witness in computer disputes and litigation Dr Castell has acted in more than 100 major cases worldwide over the past 15 years, including the largest and longest software development contract actions to have reached trial in the English High Court, and one of the most technologically complex cases to be heard in the Sydney Commercial Court. He is also experienced as a (CEDR-trained) Mediator, an ICC Arbitrator, and Expert Determiner.**

Dr Stephen Castell, Chairman
CASTELL Consulting
Tel: +44 1621 891 776
Mob: +44 7831 349 162
Fax: +44 1621 892 553
Email: cstll01@attglobal.net
PO Box 334, Witham,
Essex CM8 3LP, United Kingdom

www.CASTELLConsulting.com

European Representative Office:
Angel Enterprises
6, rue Massenet, 06000 Nice, France
Tel & Fax: +33 (0)4 93 16 97 05

Notes

[1] The following are just some of the many assignments in which *CASTELL Consulting* has been involved:

UK - longest computer software English High Court Trial: *GEC Marconi vs. London Fire and Civil Defence Authority* (1991-92).

UK - largest outsourcing/software development contract English High Court case: *Airtours vs. EDS*, £200m+ claim; £50m+ counter-claim (2001).

Court of Sessions, Edinburgh, Scotland – Materials and Manufacturing software contract litigation (1999-2003).

UK Arbitrations – high-profile cases re (1) Real-Time Mobilising System on behalf of Metropolitan Ambulance Service (1993); and (2) Courts System on behalf of Central Government Department (1994-95).

UK – High Court Litigation: Army Personnel Records, Document Management/Workflow System dispute, *Ministry of Defence vs. BT Syntegra* (1998-2000).

UK – International Chamber of Commerce (Paris) Arbitration, London: UK and German disputants – Arbitrator in a 3-man Arbitral Tribunal (2000-2001).

France, and Melbourne, Australia – on behalf of major French traffic engineering and electronics company, and International Joint Venture Consortium, prepared a Preliminary Expert's Report in connection with electronic tolling system software for Australia's largest road infrastructure project (2000-2002).

Sydney, Australia – one of the most technically complex cases in the Sydney Commercial Court: prepared Expert's Reports using specially-developed *Forensic Systems Analysis* techniques to assess quality of 6-module ERP software suite for 19 universities - *CHA vs. Unipower* (1997-98).

Dublin, Ireland – ERP/Manufacturing software contract litigation (2000).

Milan, Italy – Health Information Systems project termination (200-2001).

Riga, Latvia – Retail Banking Systems dispute (2002).

UK - IT Outsourcing Benchmarking: definitive case in connection with leading *C&W vs IBM* High Court Action over methodologies used and results found in an Outsourcing Value Analysis project as undertaken by a major IT Benchmarking Consultancy (2004).

UK and USA – computers forensics: determining the presence of, and retrieving, electronic documents from computer PC harddisks and server systems as evidence in e.g. criminal and fraud legal proceedings (2003-2004).

UK – Telecoms: on behalf of *British Telecom*, determining the validity of telephone subscriber numbers and usage in regard to multi-million pound contract dispute with local retail telephony Service Provider (2004).

USA – Telecoms and Broadcasting - Patents and Intellectual Property actions: e.g. databroadcasting and billing systems (2003-2004).

In many of these cases, we have worked very successful with indemnity insurers (for example *Hiscox*, the world's leading IT&T and media insurers) and have frequently achieved rapid, highly cost-effective settlement of disputes upon exchange of our Expert's Reports.

[2] See for example:

a. '*Give the IT expert a chance, please… Dr Stephen Castell offers a helpful mini-handbook of suggestions to solicitors for instructing him as computer expert witness in complex software implementation contract disputes…*', *The Barrister*, June 2004.

b. 'Recognising early the signature of *DISPRO* – The IT *DIS*aster *PRO*ject', Stephen Castell, paper presented at the *CLT* London Conference '*Buying Computer Systems*', held 12 November 2003. Stephen Castell conceived and Co-Chaired this Conference (with John Boyd QC), which included a *Mock Trial*, and which dealt *inter alia* with how to:

- Assure quality software supply and delivery processes
- Avoid the pitfalls of systems projects failures
- Resolve disputes between customers and suppliers
- Assess objectively the quality and fitness for purpose of the system supplied
- Learn from the mistakes of others
- Measure, secure and protect IT investment.

c. '*Split your outsourcing contracts to guard against legal disputes*', John Kavanagh, *Computer Weekly*, 14 October 2003, Career moves, British Computer Society, page 76.

d. '*How to avoid project disasters*', Stephen Castell, *Butler Group Review*, Management Matters, September 2003, page 23.

e. '*Low-stress development. Contracting out software development – on its own or as part of a wider IT management agreement – demands close attention to many specific issues. Stephen Castell shares some lessons that many companies are learning the hard way*', British Computer Society, *The Computer Bulletin*, September 2003, pages 24-25.

f. '*The role of the IT expert witness in software and systems development/implementation contract disputes and litigation*', Stephen Castell, Expert Witness Institute *Newsletter*, Summer 2003, pages 1-3 (synopsis of a talk given in London to the *Association of Independent Computer Specialists*, 10 October 2002). [Published also in the *Computer Law & Security Report*, Vol. 19, No. 3, 2003, pages 228-231].

g. '*Key role of expert witnesses*', Stephen Castell, *The Times* Law Letters, 25 March 2003, page 10.

h. '*IT Expert and Counsel in Computer Software Disputes – Professionals in Harmony*', Stephen Castell, *The Barrister*, 1 October 2001, pages 6-7. [Published also in *Information Security Bulletin*, Vol, 7, Issue 11, November 2002, pages 39-40, as '*IT Expert And Legal Counsel – Ad Idem in Computer Software Disputes*'].

i. '*Quick'n'dirty resolves it. Grania Langdon-Down assesses the worth of expert determination*', *The Times*, 15 February 2000: "Stephen Castell, an IT consultant who has acted as expert witness, mediator and arbitrator, said expert determination might be a way of resolving disputes more quickly and simply than waiting for them to settle just before court…".

j. '*Reality Bytes*', Matt Barnard, Special Report: Expert Witnesses, *The Lawyer*, 22 November 1999, pages 29-30: "… '… software … [is] going to become an increasingly common area of dispute…. You could almost imagine a whole new industry of technological dispute resolution growing up,' says Castell".

k. '*Software: hard evidence. Stephen Castell says the key to unravelling computer software disputes is understanding what is meant by "software quality"*', Experts: IT Systems, Stephen Castell, *Legal Week*, 29 April 1999, page 25.

l. '*The Millennium Bug: Well of course it's a Software Fault but is it Contractually a* ***Material Defect*** *?*, Stephen Castell. Presentation given at the '*Millennium Bug: Who will pay the price ?* Meeting and 'Mock Trial', co-sponsored by Berrymans Lace Mawer, Solicitors; The Institution of Electrical Engineers; and the *Law Specialist Group* of the British Computer Society, and held at the IEE, Savoy Place, London, 12 January 1999.

m. '*Are mission critical computer systems creating litigation ?*, Stephen Castell, *IN BRIEF*, November/December 1998, page 45.

n. Letters published in the *Law Society's Gazette*:
- 'Y2K FALLACY', Stephen Castell, *LSG*, 20 January 1999.
- 'BUG BOTHER', Jonathan Mounteney, *LSG* 6 January 1999;
- 'BUGGED BY *BOLITHO*', Stephen Castell, *LSG* 16 December 1998;
- 'ERADICATING THE BUG', Stephen Castell, *LSG* 30 September 1998;

o. '*Computer Litigation – Somewhere in Your Future ?*, Stephen Castell, *Information Security Bulletin*, Volume 3, Issue 4, June 1998, Page 23ff.:
"... clear computer judgments, founded on a proper understanding by the Court of established professional and technical principles and computer system development methodologies, are arguably badly needed as precedents by the computer software industry and its legal advisers alike..."

p. '*Ironing out the bugs. When it comes to solving computer disputes, an independent expert may well prove more effective than a heavily armed lawyer*', Stephen Castell, IT Chapter Section of *Charter*, the Journal of the Institute of Chartered Accountants in Australia, November 1997:
"Those responsible for keeping company... records, including software development project management and technical documentation (... not forgetting... e-mails and other Internet/Intranet documents), should always bear in mind the potential need for their disclosure in a legal action..."

q. '*Computer Litigation – an expert speaks*', Stephen Castell, *IN BRIEF*, June 1997:
"There has been a dramatic increase in the number of computer-related disputes ending up in court...".

r. '*COURT IN THE ACT*', *Computer Weekly*, 13 February 1997:
"Who is to blame for the year 2000 problem and why should lawyers really care when they are likely to get paid huge sums for arguing the toss ? *Shan Kelly* reports on the battle lines being drawn over year 2000 liability... who exactly is responsible for the fact that most computers don't understand four-digit dates ? Who is to blame for most companies being forced to spend the equivalent of one-and-a-half times their annual IT budget explaining to computerised systems that the year 2000 comes after 1999 and that it is a leap year ? The British Computer Society (BCS) Law Specialist Group, which now has more than 400 members, has been giving detailed consideration to the issue...".

s. '*Seeking after the truth in computer evidence: any proof of ATM fraud ?*, Stephen Castell, *Computer Bulletin*, December 1996, pp. 17-19:
"... my study on computer evidence for the CCTA... highlighted the need for computer systems and operational practices properly capable of forensic scrutiny, delivering undoubted evidential reliability... every criminal trial which seeks to rely on computer evidence should first be a trial of the computer systems from which evidence is to be derived."

[3] The name *Forensic Systems Analysis* was first coined by Larry Traynor, a Senior CASTELL Associate Consultant working as part of the CASTELL Consulting expert team instructed during 1991-92 on the *GEC Marconi Ltd –v– London Fire and Civil Defence Authority* case [1989-ORB-No. 1208]. (*Forensic Systems Analysis* should not be confused with '*computer forensics*' a more recent, and non-proprietary, expression, which occurs in relation to 'low-level' technical tools and techniques for obtaining computer evidence from e.g. computer disks in – principally – criminal cases.)

Annex A

CASTELL Pro-Forma for *Scott Schedule (of Software Defects)*

Part A of Schedule
Identification of each Item in the Schedule

Column in Schedule	Column Heading and Further Explanation	Short-Form
1. A1	Item Number *Sequential reference number for each item*	'Item'
2. A2	Brief Description/Title	'Title'
3. A3	*TIR* Reference Number	'*TIR* No.'

Part B of Schedule
Details of Software Requirements and Software Defects for each Item

Column in Schedule	Column Heading and Further Explanation	Short-Form
4. B1	Detailed Description of Software Requirements *Here, the Customer sets out to the fullest extent possible a clear and complete presentation of what precisely it says were the contractual software requirements for this Item*	'Requirements'
5. B2	*Statement of Requirements* Reference(s)	'*SoR* Refs'
6. B3	Other Specification or Contractual Document Reference(s)	'Other Refs'
7. B4	Details of Acceptance Testing	'Tests Done'
8. B5	Acceptance Test Material Reference(s)	'Test Refs'
9. B6	Detailed Description of Defects [*For this Item, what does the Customer allege were the defects in respect of which the software allegedly failed testing ?*] *Here, the Customer sets out to the fullest extent possible a clear and complete presentation of the alleged defects in this Item, why they are considered by the Customer to be so, any exhibits arising from its testing (e.g. screen printouts) which it says demonstrate the defects, etc etc. This should include specific program/software module references etc etc*	'Defects'

Part B of Schedule ***(continued)***
Details of Software Requirements and Software Defects for each Item

Column in Schedule	Column Heading and Further Explanation	Short-Form
10. B7	Type of Acceptance Criteria Failed, for example: 'F' - Functionality 'O' - Operability 'P' - Performance	'Criteria'
11. B8	Date Defect Discovered by the Customer	'Date Found'
12. B9	Date Defect Notified by Customer to the Supplier	'Date Notified'
13. B10	Supplier's Response and Comments regarding Software Requirements and Defects	'Supp. Response'
14. B11	Supplier's Documentary References regarding Software Requirements and Defects	'Supp. Docs'
15. B12	Details of any legal issues to be determined regarding Software Requirements and Defects	'Legal Issues 1'

Part C of Schedule
Details of Materiality of Software Defects

Column in Schedule	Column Heading and Further Explanation	Short-Form
16. C1	Materiality: Consequential Effect [*To be considered a **material defect** an alleged software defect should unequivocally meet a generally accepted test, viz: It must have a **large consequential effect** if left unfixed **and** it must take, or have taken, a **long time to fix** (both must be true)*] *Here, the Customer sets out what it believed was or would have been the consequential effect of the Item if left unfixed*	'Cust. - Effect'
17. C2	Supplier's Response and Comments regarding Materiality: Consequential Effect	'Supp. - Effect'
18. C3	Materiality: Time to Fix [*To be considered a **material defect** an alleged software defect should unequivocally meet a generally accepted test, viz: It must have a **large consequential effect** if left unfixed **and** it must take, or have taken, a **long time to fix** (both must be true)*] *Here, the Supplier sets out the time it estimated it would have taken, or states the time it actually did take, to fix the Item.*	'Supp.– Fix Time'
19. C4	Customer's Response and Comments regarding Materiality: Time to Fix	'Cust. – Fix Time'

Part C of Schedule ***(continued)***
Details of Materiality of Software Defects

Column in Schedule	Column Heading and Further Explanation	Short-Form
20. C5	Materiality: Details of Workaround (if any) Available During Acceptance Test: **Customer's View** [*If during an Acceptance Test there is available a workaround to an encountered software defect, it is generally held that (since the point of an Acceptance Test is to try to accept, not reject, the software being tested) the defect should not be considered as a material defect. The defect may still however need to be fixed before final implementation of the software if the workaround is not, in the opinion of the customer, an acceptable permanent alternative*] *Here, the Customer gives its view as to whether or not there was any workaround available for this Item; and, if so, what it was; and whether or not it was acceptable to the Customer as a permanent alternative (and if not, why not)*	'W/A – Cust.'
21. C6	Materiality: Details of Workaround (if any) Available During Acceptance Test: **Supplier's View** [*If during an Acceptance Test there is available a workaround to an encountered software defect, it is generally held that (since the point of an Acceptance Test is to try to accept, not reject, the software being tested) the defect should not be considered as a material defect. The defect may still however need to be fixed before final implementation of the software if the workaround is not, in the opinion of the customer, an acceptable permanent alternative*] *Here, the Supplier gives its view as to whether or not there was any workaround available for this Item; and, if so, what it believes it was*	'W/A – Supp.'
22. C7	Details of any legal issues to be determined regarding Materiality of Software Defects	'Legal Issues 2'

Part D of Schedule
Determination of the Item

Column in Schedule	Column Heading and Further Explanation	Short-Form
23. D1	New Evidence: Results of Repeat Tests [*For this Item, what tests did the Customer conduct to establish its allegations ? If the Court wishes to re-visit such tests, what details need to be available ?*] *Details of any Repeats of the Test(s) done by the Customer, ordered by/carried out for the Court to assist in its determining the Item, together with the Results thereof*	'Repeat Tests'

Part D of Schedule ***(continued)***
Determination of the Item

Column in Schedule	Column Heading and Further Explanation	Short-Form
24. D2	New Evidence: Results of New Tests *Details of any New Test(s) ordered by/carried out for the Court to assist in its determining the Item, together with the Results thereof*	'New Tests'
25. D3	New Evidence: Customer's Independent Expert's Opinion on this Item	'Cust. Expert'
26. D4	New Evidence: Supplier's Independent Expert's Opinion on this Item	'Supp. Expert'
27. D5	Court's Determination: For this Item, were the Software Requirements as detailed by the Customer actually contractual software requirements? 'Y' - 'Yes' 'N' - 'No'	'Court 1'
28. D6	Court's Determination: For this Item, did the Customer's tests as implemented for the software utilise the Acceptance Test Material as accepted by the Supplier? 'Y' - 'Yes' 'N' - 'No'	'Court 2'
29. D7	Court's Determination: For this Item, were the Supplier's tests appropriate ? 'Y' - 'Yes' 'N' - 'No'	'Court 3'
30. D8	Court's Determination: For this Item, were the Customer's tests conducted appropriately ? 'Y' - 'Yes' 'N' - 'No'	'Court 4'
30. D9	Court's Determination: For this Item, did the Customer's tests properly identify the alleged defect, ie were the consequent results correct and presented such that the Supplier could properly address the alleged defect ? 'Y' - 'Yes' 'N' - 'No'	'Court 5'

Part D of Schedule ***(continued)***
Determination of the Item

Column in Schedule	Column Heading and Further Explanation	Short-Form
31. D10	Court's Determination: a. For this Item, was the alleged defect actually a defect ? 'Y' - 'Yes' 'N' - 'No' b. If so, was the defect a material defect ? 'Y' - 'Yes' 'N' - 'No'	'Court 6a' 'Court 6b'
32. D11	Details of any legal issues to be determined regarding Determination of the Item	'Legal Issues 3'
33. D12	Judge: Comments and Decision	'Judgment'

Annex B

CASTELL Pro-Forma for *Scott Schedule (of Software Extras)*

Part A of Schedule
Identification of each Item in the Schedule

Column in Schedule	Column Heading and Further Explanation	Short-Form
1. A1	Item Number *Sequential reference number for each item*	'Item'
2. A2	Brief Description/Title	'Title'
3. A3	Project Reference Number	'Proj. No.'
4. A4	Module/Program Number *Module/program number as appropriate to identify the area of the applications software under review*	'Module'

Part B of Schedule
Details of the Extra, the Original Requirement and Delivery of the Extra

Column in Schedule	Column Heading and Further Explanation	Short-Form
5. B1	Customer's Definition of the New Requirement *Definition of the new, modified or extra requirement, and references to the source of the requirement*	'Req-Cust.'
6. B2	Detailed Description of the Extra *How the requirement was implemented and any supporting document references (eg design specifications)*	'Extra'
7. B3	Support or otherwise for the original software *Support for the original software as defined by the signed-off Statement of Requirements and other relevant material*	'*SoR* Refs'
8. B4	Other Specification/Contractual References *Any further support for the original requirement (eg 'mini-specifications', correspondence etc)*	'Other Refs'
9. B5	Further Comments from the Supplier *Any further comments/references from the Supplier in support of the claim that the work and materials were in excess of those required by the Contract*	'Supp. Comm'

Part B of Schedule ***(continued)***
Details of the Extra, the Original Requirement and Delivery of the Extra

Column in Schedule	Column Heading and Further Explanation	Short-Form
10. B6	Customer's Response relating to the Item *Customer to justify any reasons it may have for the item not being classified as an extra, including references to the SoR and any other specifications or references that it believes to be appropriate*	'Cust. Resp'
11. B7	Details of the Delivery of the Item of Extra Work *Supplier to provide evidence of the date of delivery of the item*	'Supp. Del'
12. B8	Customer Response to the Delivery of the Item *Customer to identify any deficiencies or inadequacies in either the delivery or the completeness of the item*	'Cust. Del'
13. B9	Details of any legal issues to be determined regarding Determination of the Item	'Legal Issues 1'

Part C of Schedule
Determination of the Item

Column in Schedule	Column Heading and Further Explanation	Short-Form
14. C1	New Evidence: Results of Repeat Assessments [*For this Item, what assessments did the Supplier conduct to establish its claims ? If the Court wishes to re-visit them, what details need to be available ?*] *Details of any Repeats of the Assessment(s) performed by the Supplier, ordered by/carried out for the Court to assist in its determining the Item, together with the Results thereof*	'Repeat Assess'
15. C2	New Evidence: Results of New Assessments *Details of any New Assessment(s) ordered by/carried out for the Court to assist in its determining the Item, together with the Results thereof*	'New Assess'
16. C3	New Evidence: Customer's Independent Expert's Opinion on this Item	'Cust. Expert'
17. C4	New Evidence: Supplier's Independent Expert's Opinion on this Item	'Supp. Expert'
18. C5	Court's Determination: For this Item, did the Supplier carry out work and provide materials in excess of those required by the Contract ? 'Y' - 'Yes' 'N' - 'No'	'Court 1'

Part C of Schedule ***(continued)***
Determination of the Item

Column in Schedule	**Column Heading and Further Explanation**	**Short-Form**
19. C6	Court's Determination: For this Item, did the Supplier deliver the item of extra work to the Customer ? 'Y' - 'Yes' 'N' - 'No'	'Court 2'
20. C7	Details of any legal issues to be determined regarding Determination of the Item	'Legal Issues 2'
21. C8	Judge: Comments and Decision	'Judgment'

[*7,800 words approximately*]